We discuss the second application of Depth-first Search (DFS): Strongly connected components. We start with some definitions. Let $G = (V, E)$ be a directed graph. We say $C$ is a strongly connected component (SCC) of $V$ if it is a maximal set of vertices such that every two vertices $u, v \in C$ are mutually reachable: there is a path from $u$ to $v$ as well as a path from $v$ to $u$. The word “maximal” basically means for any $u \in C$ and $w \notin C$, $u$ and $w$ are not mutually searchable. Check Appendix B.
Every directed $G = (V, E)$ can be partitioned into pairwise disjoint SCCs, just like connected components in an undirected graph:

$$C_1, C_2, \ldots, C_k,$$  for some $k \geq 1$

The SCC problem is then the following:

Given a directed graph $G = (V, E)$, find its SCCs
Given $G$, let $G^T = (V, E^T)$ denote the reverse graph of $G$:

$$E^T = \{(v, u) : (u, v) \in E\}$$

It is easy to see that, by the definition of $G^T$, $v$ is reachable from $u$ in $G^T$ if and only if $u$ is reachable from $v$ in $G$. Quick question: Given the list representation of $G$, how to construct the list representation of $G^T$ in linear time?
First try: here is a straight-forward SCC algorithm using DFS:

1. pick an arbitrary vertex \( u \) from \( V \)
2. call DFS-Visit \((G, u)\) to get \( R \): vertices reachable from \( u \) in \( G \)
3. call DFS-Visit \((G^T, u)\) to get \( R' \): vertices reachable from \( u \) in \( G^T \) or equivalently, \( v \in R' \) iff \( u \) is reachable from \( v \) in \( G \)
4. output \( R \cap R' \) (show that this is the SCC that contains \( u \))
5. remove \( R \cap R' \) from \( G \) and repeat, until \( G \) is empty
However, its worst-case running time is not good. Each call to DFS-Visit costs $O(n + m)$ so the total running time is

$$O(n(n + m))$$

There are also worst-case examples to show that $\Omega(nm)$ time is necessary (try to construct one by yourself). We will show that there is actually a $O(n + m)$ linear-time algorithm for the SCC problem!!! Much more efficient.
We start with the following lemma about the SCCs of $G$:

**Lemma**

Let $C$ and $C'$ be two SCCs of $G$. If there is an edge from $C$ to $C'$ in $G$, then there is no edge from $C'$ to $C$.

Otherwise, show that $u \in C$ and $v \in C'$ are mutually reachable and thus, one can merge $C$ and $C'$ to get a larger SCC, contradiction.
This leads us to define the component graph $G_{SCC}$ of $G$: each SCC of $G$ corresponds to a vertex in $G_{SCC}$ so the vertices of $G_{SCC}$ are

$$\{C_1, \ldots, C_k\}, \quad \text{where } C_1, \ldots, C_k \text{ are the SCCs of } G$$

and $(C_i, C_j)$ is an edge in $G_{SCC}$ if there is an edge from $C_i$ to $C_j$ in $G$. Given this definition, it is easy to prove the following lemma:

**Lemma**

*The component graph $G_{SCC}$ of $G$ must be a DAG.*

Quick proof: Assume there is a cycle $C_1 C_2 \cdots C_\ell = C_1$ of length $\ell \geq 2$ in $G'$. Then it can be shown that any two vertices in $\bigcup_{i=1}^{\ell} C_i$ are indeed mutually reachable in $G$ and thus, one can merge $C_1, \ldots, C_{\ell-1}$ to obtain an even larger SCC, contradiction.
We know that as a DAG, $G_{SCC}$ must have at least one source (a vertex with no incoming edges) and at least one sink (a vertex with no outgoing edges). We call an SCC $C$ of $G$ a source (sink) SCC if $C$ corresponds to a source (sink) vertex in $G_{SCC}$. So $C$ is a sink SCC of $G$ if there is no edge from $C$ to other SCCs of $G$. The following lemma is the key idea behind the linear-time algorithm:

**Lemma (3)**

Let $C$ be any sink SCC of $G$ and $u$ be any vertex in $C$. Then $C$ is exactly the set of vertices reachable from $u$ in $G$. Therefore, to compute $C$, one only needs to compute the set of vertices reachable from $u$ in $G$ by making a call to DFS-Visit($G$, $u$).
Before proving Lemma 3, it suggests the following algorithm:

1. find a vertex $u \in V$ in a sink SCC of $G$
2. call DFS-Visit ($G, u$) to get $R$: vertices reachable from $u$
3. output $R$, the SCC that contains $u$ according to Lemma 3
4. remove $R$ from $G$ and repeat, until $G$ is empty

Clearly the main problem left is how to find a vertex $u$ in a sink SCC of $G$. Another subtle problem is, after removing a SCC from $G$, how to find a vertex in a sink SCC of the remaining graph.
Proof of Lemma 3: Let $R$ denote the set of vertices reachable from $u$ in $G$. By definition we have $C \subseteq R$ because $v \in C$ means not only $v$ is reachable from $u$ but also $u$ is reachable from $v$. We need to show $C = R$ when $C$ is a sink SCC of $G$. Show that if $v \notin C$, then $v$ is not reachable from $u$ in $G$, by using the assumption that $C$ is a sink SCC.
Now we discuss how to find a vertex in a sink SCC of $G$. Note that $C$ is an SCC of $G$ if and only if it is an SCC of $G^T$; $C$ is a sink SCC of $G$ if and only if it is a source SCC of $G^T$. So it suffices to find a vertex in a source SCC of $G^T$ efficiently. The following lemma shows how: We start by running DFS on $G^T$! Upon termination, let $u.f$ denotes the finish time of $u$ in DFS($G^T$). (We use $u.f$ to denote the finish time in DFS($G^T$) in the rest of the note.)

**Lemma (4)**

*The vertex $u$ with the largest finish time $u.f$ must belong to a source SCC of $G^T$ and thus, a sink SCC of $G$.***
Lemma 4 is a corollary of the following stronger lemma (why?): Given an SCC $C$ of $G^T$ (and $G$ as well), we use $f(C)$ to denote

$$f(C) = \max_{u \in C} \{ u.f \}$$

the maximum finish time of vertices in $C$. Again, remember that $u.f$ denotes the finish time of $u$ in DFS($G^T$). (Note the subtle difference between the presentations of the note and textbook.)

**Lemma (5)**

Let $C$ and $C'$ be two SCCs of $G^T$ (and $G$ as well). If there is an edge from $C$ to $C'$ in $G^T$, then we must have $f(C) > f(C')$.  

Introduction
Consider the following two cases: In $\text{DFS}(G^T)$, $C$ is visited before $C'$ or $C'$ is visited before $C$. For the first case, let $u \in C$ be the first vertex $\text{DFS}$ discovers among $C \cup C'$. Then at the time when $u$ is discovered, all vertices in $C \cup C'$ are white and thus, there is a white path from $u$ to every vertex in $C \cup C'$ (why? use the assumption that there is an edge from $C$ to $C'$ in $G^T$). Therefore, by the White-Path theorem, all vertices in $C \cup C'$ are descendants of $u$ in the depth-first forest and thus, by the Parenthesis lemma $u$ has the largest finish time and $f(C) > f(C')$ because $u \in C$. 
The other case: Assume $u \in C'$ is the first vertex DFS discovers among $C \cup C'$. At the time when $u$ is discovered, all vertices in $C'$ are white and thus, there is a white path from $u$ to every vertex in $C'$. Therefore, by the White-Path theorem, $u$ has the largest finish time in $C'$ and $f(C') = u.f$. However, every vertex $v \in C$ is not reachable from $u$ in $G^T$ (why?). Because $v \in C$ is not discovered at the time $u.d$, it remains white at the time $u.f$ (why? use the White-Path theorem). Therefore, $v.d > u.f$ and $f(C) > f(C')$. 

From Lemma 5, we can simply call $\text{DFS}(G^T)$ to find the vertex $u$ with the largest finish time $u.f$. It must belong to a source SCC of $G^T$ and thus, a sink SCC of $G$. By Lemma 3, $\text{DFS-Visit}(G, u)$ returns the SCC $C$ that contains $u$. But how do we continue after removing $C$ from $G$? Do we need to call DFS on the new graph?
No! Here is the technically most important idea in the linear-time algorithm for SCC. After we found the first SCC and delete it from $G$, it can be shown that the vertex $v$ with the largest finish time $v.f$ from DFS($G^T$) among the remaining vertices must belong to a sink SCC $C'$ of the remaining graph, denoted by $G'$. Therefore, we can just call DFS-Visit($G'$, $v$) to get the SCC $C'$ that contains $v$, which is the second SCC of $G$ we find simply because an SCC of $G'$ is also an SCC of $G$. We can repeat to find the third SCC of $G$, the fourth, and so on. Note that we only call DFS($G^T$) once.
Why does $v$, the vertex with the largest finish time $v.f$ in the remaining graph $G'$, after deleting the first SCC $C$ we found, belong to a sink SCC $C'$ of $G'$? This follows from Lemma 5: If there is another SCC $C^*$ in $G'$ with an edge from $C'$ to $C^*$ in $G'$, then there is an edge from $C^*$ to $C'$ in $G^T$ and thus,

$$f(C^*) > f(C')$$

This contradicts the assumption that $v \in C'$ has the largest finish time among vertices in $G'$. Similarly, by induction one can show that after removing the second SCC, the vertex with the largest finish time in the remaining graph belongs to a sink SCC of the remaining graph, and so on.
To summarize, here is the linear-time algorithm for SCC:

1. construct the adjacency list representation of $G^T$ from $G$
2. call DFS($G^T$) to get a reordering $S$ (a linked list) of the vertices $V$ with their finish times sorted from large to small
3. while $G$ and $S$ are not empty do
   4. let $u$ be the first vertex in $S$
   5. call DFS-Visit($G, u$) to get $R$: vertices reachable from $u$
   6. $R$ must be the SCC that contains $u$
   7. remove $R$ from $G$ and $S$
Both line 1 and line 2 can be done in time $O(n + m)$ (for line 2, recall the linear-time topological sort algorithm). To see why the while-loop takes time $O(n + m)$, note that essentially it is DFS($G$) with vertices in the for-loop of DFS($G$) ordered as in $S$. 